Date=09/10/2020
Lecture By=Arkesh Jaiswal
Subject ⇒Javascript Objects

IN PREVIOUS LECTURE **(QUICK RECAP) Date-08/10/2020**	**In Today's Lecture (Overview)**
• JavaScript Objects • Object • Properties • Methods • Object Properties • Accessing Object Properties • You can access object properties in two ways: • The this Keyword • Accessing Object Methods • Questions For self Practice • Resource for the lecture	• JavaScript Closures • Global Variables • JavaScript Nested Functions • Questions For Self Practice / Assignment/CC For the day • Resources For the Lecture • Youtube Tutorial

JavaScript Closures

JavaScript variables can belong to the local or global scope.

Global variables can be made local (private) with closures.

Global Variables

A `function` can access all variables defined inside the function, like this:

Example

```
function myFunction() {
  var a = 4;
  return a * a;
}
```

But a `function` can also access variables defined outside the function, like this:

Example

```
var a = 4;
function myFunction() {
  return a * a;
}
```

In the last example, a is a global variable.

In a web page, global variables belong to the window object.

Global variables can be used (and changed) by all scripts in the page (and in the window).

In the first example, a is a local variable.

A local variable can only be used inside the function where it is defined. It is hidden from other functions and other scripting code.

Global and local variables with the same name are different variables. Modifying one, does not modify the other.

A Counter Dilemma

Suppose you want to use a variable for counting something, and you want this counter to be available to all functions.

You could use a global variable, and a `function` to increase the counter:

Example

```
// Initiate counter

var counter = 0;

// Function to increment counter

function add() {

  counter += 1;

}

// Call add() 3 times

add();

add();

add();

// The counter should now be 3
```

There is a problem with the solution above: Any code on the page can change the counter, without calling add().

The counter should be local to the `add()` function, to prevent other code from changing it:

Example

```
// Initiate counter
```

```
var counter = 0;

// Function to increment counter

function add() {

  var counter = 0;

  counter += 1;

}

// Call add() 3 times

add();

add();

add();

//The counter should now be 3. But it is 0
```

JavaScript Nested Functions

All functions have access to the global scope.

In fact, in JavaScript, all functions have access to the scope "above" them.

JavaScript supports nested functions. Nested functions have access to the scope "above" them.

In this example, the inner function `plus()` has access to the `counter` variable in the parent function:

Example

```
function add() {
```

```
  var counter = 0;

  function plus() {counter += 1;}

  plus();

  return counter;

}
```

JavaScript Closures

Remember self-invoking functions? What does this function do?

Example

```
var add = (function () {

  var counter = 0;

  return function () {counter += 1; return counter}

})();

add();

add();

add();

// the counter is now 3
```

Example Explained

The variable `add` is assigned the return value of a self-invoking function.

The self-invoking function only runs once. It sets the counter to zero (0), and returns a function expression.

This way add becomes a function. The "wonderful" part is that it can access the counter in the parent scope.

This is called a JavaScript closure. It makes it possible for a function to have "private" variables.

The counter is protected by the scope of the anonymous function, and can only be changed using the add function.

Questions For Self Practice / Assignment/CC For the day

https://au-assignment.s3.amazonaws.com/class-278175a9-6b4f-4acf-a42c-145fddff7930.pdf
https://au-assignment.s3.ap-south-1.amazonaws.com/cc-574df17c-57e3-4635-b94c-b11a32ac4a32.pdf

Resources For the Lecture

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures

Youtube Tutorial

https://www.youtube.com/watch?v=71AtaJpJHw0